

Figure 4: Interaction cases in closed-loop results (part 1).

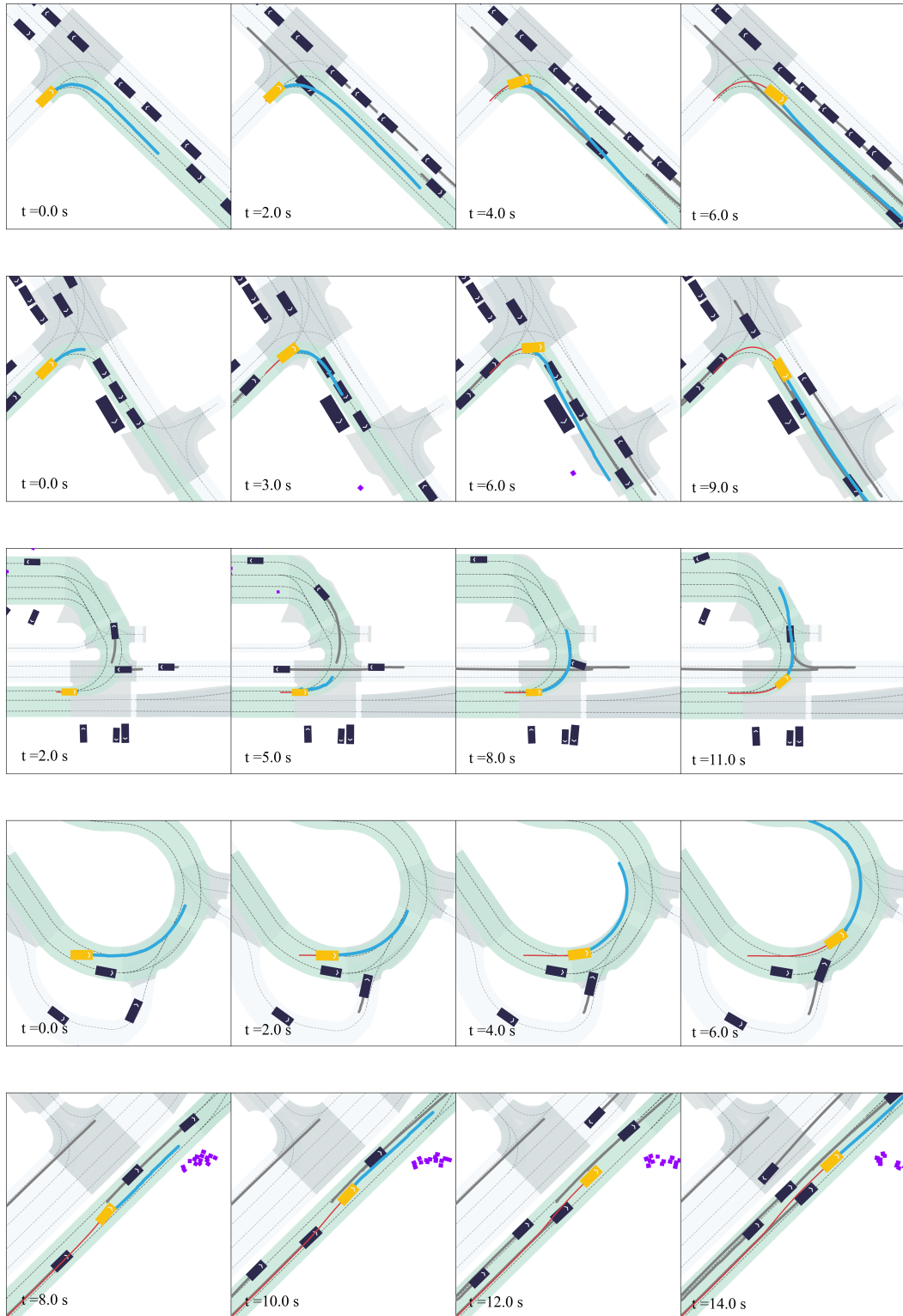


Figure 4: Interaction cases in closed-loop results (part 2).

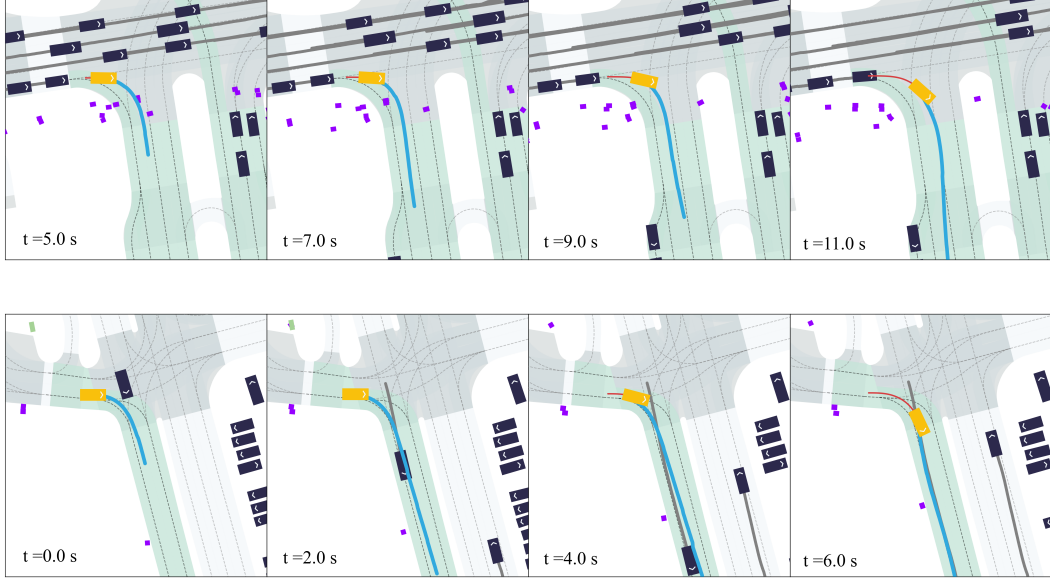


Figure 4: Interaction cases in closed-loop results (part 3).

## B Experimental Details

The training is conducted on 8 NVIDIA A6000 GPUs, using the 1M training data split from nuPlan. The model is trained for over 200 epochs with a batch size of 2048. We used AdamW optimizer for training, and the learning rate is set to be  $5 \times 10^{-4}$ . In addition, we used exponential moving average (EMA) to stabilize the training process, with 0.999 weight decay. During inference, we used a simple midpoint solver to solve the flow ODE, with only four steps of ODE simulation. The frequency of model inference is approximately 12Hz. The details for training and inference can be found in Table. 6

Table 6: Hyperparameters of *Flow Planner*

Type	Parameter	Symbol	Value
Training	Num. neighboring vehicles	-	32
	Num. past timestamps	$T$	21
	Dim. neighboring vehicles	$D_{\text{neighbor}}$	11
	Num. lanes	-	70
	Num. points per polyline	-	20
	Dim. lanes vehicles	$D_{\text{lane}}$	12
	Num. navigation lanes	-	25
	Num. encoder block	-	3
	Num. decoder block	-	4
	Dim. encoder hidden layer	-	192
	Dim. decoder hidden layer	-	256
	Num. multi-head	-	8
	Len. trajectory segment	$L_{\text{seg}}$	20
	Len. trajectory overlap	$L_{\text{overlap}}$	10
Inference	Flow path	-	Conditional OT
	Temperature	-	1.0
	Flow ODE simulation step	-	4

## C Benchmarks and Baselines

**Benchmark Details.** We used the Val14, Test14-random and Test14-hard benchmarks for evaluation, as stated in Section 4. Each of the benchmarks contains the 14 scenarios specified by the nuPlan leaderboard respectively. The models are tested in two modes: non-reactive and reactive. In the non-reactive settings, the neighboring vehicles and other traffic participants strictly follow their trajectories in the record and will not react to the ego vehicle; on the contrary, in the reactive mode, the neighboring vehicles are controlled by the rule-based IDM [48]. However, the reaction generated by IDM can sometimes be suboptimal and lead to unnatural behavior of surrounding vehicles.

**Baseline Reproduction.** We followed [55] to reproduce the baselines. For *Diffusion Planner*, we used the official implementation to reproduce the results, and the model is trained for 500 epochs, which is more than twice the number of *Flow Planner*.

## D Limitation & Discussion & Future Work

The inference speed of *Flow Planner* currently represents its main limitation. While our shift from diffusion to flow matching has improved sampling efficiency, the system achieves only a speed marginally faster than 10 Hz on an A6000 GPU, where 10 Hz is the requirement for industrial deployment [17]. The main reason leading to low sampling efficiency is that the spatiotemporal self-attention mechanism remains computationally intensive. Future work will explore acceleration techniques like the shortcut model [18] to address these constraints without sacrificing planning quality. Besides, for the interactive behavior modeling, we still use an implicit design to enhance this capability. The whole method is still an imitation learning-based method, facing the problem of data quality and data forgetting issues. Maybe we can use the reinforcement learning approach to further enhance the capability of interactive behavior modeling, but there is still a long way to go to tackle problems for real-world vehicle implementation.

## E Implementations of Training

**Data Preprocessing.** A scenario in nuPlan contains versatile scene information, whereas not all of it is used for model prediction. In our method, the model takes the lanes, neighboring agents (vehicles, pedestrians, cyclers etc.), navigation information and static objects from the scenario as input. These scenario inputs are collected into tensors once and for all, and can be reused during training.

- *Lanes*: a lane is represented by a sequence of 20 points sampled from the past 2 seconds at 10 Hz, each of which contains the coordinates of the lane center line and lane boundary line, as well as the connecting vector of center line and lane traffic signal; up to 70 lanes are fed into the model;
- *Agents*: the nearest 32 agents to the ego vehicle is encoded as the model input, and each agent contains the historical trajectories in the past 2 seconds at 10Hz, and states in the current frame;
- *Navigation*: represented by 5 lanes along the assigned direction, saved in the same form as lanes;
- *Static Objects*: the static obstacles in the scenario, represented by a vector containing the position in the ego centric coordinates at the current frame and the specific type of the obstacle;

**Transformation and Normalization.** The original data in nuPlan is recorded in the world coordinates, in which the starting point of the ego vehicle can appear in any place, introducing unnecessary complexities for training. Therefore, to unify the numerical value of model input, we first transform the entire scenario into the ego-centric front-left-up coordinate, in which the state of the ego vehicle at the first frame of each scenario is  $(X = 0, Y = 0, \theta = 0)$ , and the coordinates of other scenario instances are transformed accordingly. Specifically, given the world coordinates of the ego vehicle  $(X, Y, \theta)$ , the transformation from the original coordinates  $(x, y)$  to the transformed coordinates  $(x', y')$  can be formulated as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta & -X \cos \theta - Y \sin \theta \\ -\sin \theta & \cos \theta & X \sin \theta - Y \cos \theta \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (10)$$

In addition, the original scale of coordinates varies in different scenario. To ensure the numerical stability, we normalized the coordinates and trajectories with statics from training data via z-score



575 along the x-axis and scaled along the y-axis [9, 55]. The model directly predicts the normalized  
576 future trajectory, and it is then denormalized into the scale of the real-world trajectory.

577 **Data Augmentation.** Following previous practice [55], we applied data augmentation for more  
578 robust generation. The initial states, including coordinates, velocities and heading angle, of the ego  
579 vehicle are first perturbed with random offsets. Then the perturbed initial states and the states at the  
580 20-th frame are used as the boundary condition to solve a new quintic polynomial to replace the first  
581 20 frames of the original ground truth trajectory. Intuitively, this augmentation forces the model  
582 to output a feasible future trajectory even when the ego vehicle is not properly placed on the road,  
583 enabling self-correction when the vehicle is driving off the road.